Further SRM v2.2 Proposed Changes

Compiled by Alex Sim at LBNL

Inclusive of SRM v2.2 draft 3 by
Timur Perelmutov at FNAL

With contributions from
Jean-Philippe Baud and Maarten Litmaath at CERN
and Arie Shoshani at LBNL

May 17, 2006

Abstract

In regards to recent discussion with WLCG Data Management Coordination Group for the requirements of the LHC experiments of the Grid Storage Interfaces, we propose the changes to SRM v2.1.2 specification.
This proposed SRM version 2.2 changes extend and update the current SRM v2.1.2 specification. When accepted, the changes will be consolidated into the SRM specification to be a part of SRM v2.2 specification.

Proposed V2.2 Changes

Changes from the previous doc version RC1:

1. srmPing returns SRM version number instead of boolean.
2. In srmLs, comments added for TUserID in TUserPermissions and TGroupID in TGroupPermission that they may represent DN and/or VOMS role, instead of unix style login name.
3. In srmTransferProtocolInfo, srmPing and for additional information for transfer protocols of TURL, a paired structure is returned as TExtraInfo (page 11). As the result, a string type of TTransferProtocolInfo is removed.

1. In the Space Reservation Functions:

Summary:

- Space types (TSpaceType) are removed.
- Retention policy is introduced as a way of indicating quality of the space where files are located.
- Access latency is also introduced to describe how latency of files is improvable.

- srmReserveSpace is now asynchronous, and srmGetStatusOfReserveSpace is introduced for checking status of the asynchronous srmReserveSpace.
- TMetaDataSpace includes retention policy information instead of previous space types, and file locality to indicate the current location of the file.
- srmChangeRetentionPolicy is introduced to change a retention policy of files. Since it may take a long time to complete the request, it may be an asynchronous operation, and srmGetStatusOfChangeRetentionPolicy is introduced.
- srmExtendFileLifetimeInSpace is added to extend lifefime for all files in a space that is associated with a space token.

Details:

enum **TRetentionPolicy** { REPLICA , OUTPUT , CUSTODIAL }

- Quality of Retention (Storage Class) is a kind of Quality of Service. It refers to the probability that the storage system lose a file. Numeric probabilities are self-assigned.
    - Replica quality has the highest probability of loss, but is appropriate for data that can be replaced because other copies can be accessed in a timely fashion.
    - Output quality is an intermediate level and refers to the data which can be replaced by lengthy or effort-full processes.
    - Custodial quality provides low probability of loss.
- The type will be used to describe retention policy assigned to the files in the storage system, at the moments when the files are written into the desired destination in the storage system. It will be used as a property of space allocated through the space reservation function. Once the retention policy is assigned to a space, the files put in the reserved space will automatically be assigned the retention policy of the space. The assigned retention policy on the file can be found thought the TMetaDataPathDetail structure returned by the srmLs function.

enum **TAccessLatency** { ONLINE, NEARLINE }

- Files may be Online, Nearline or Offline. These terms are used to describe how latency to access a file is improvable. Latency is improved by storage systems replicating a file such that its access latency is online.
    - The ONLINE cache of a storage system is the part of the storage system which provides file with online latencies.
    - ONLINE has the lowest latency possible. No further latency improvements are applied to online files.
    - NEARLINE file can have their latency improved to online latency automatically by staging the file to online cache.
    - For completeness, we also describe OFFLINE here.
    - OFFLINE files need a human to be involved to achieve online latency.
    - For the SRM we only keep ONLINE and NEARLINE.

- The type will be used to describe a space property that access latency can be requested at the time of space reservation. The content of the space, files may have the same or "lesser" access latency as the space.

```
typedef     struct {
                TRetentionPolicy        retentionPolicy,
                TAccessLatency          accessLatency
            } TRetentionPolicyInfo
```

- TRetentionPolicyInfo is a combined structure to indicate how the file needs to be stored.
- When both retention policy and access latency are provided, their combination needs to match what SRM supports. Otherwise request will be rejected.

srmReserveSpace

This function is used to reserve a space in advance for the upcoming requests to get some guarantee on the file management. Asynchronous space reservation may be necessary for some SRMs to serve many concurrent requests.

In:	TUserID	authorizationID,
	String	userSpaceTokenDescription,
	TRetentionPolicyInfo	preferredRetentionPolicyInfo,
	TSizeInBytes	sizeOfTotalSpaceDesired,
	TSizeInBytes	sizeOfGuaranteedSpaceDesired,
	TLifeTimeInSeconds	lifetimeOfSpaceToReserve,
	Int []	expectedFileSize,
	TStorageSystemInfo	storageSystemInfo
Out:	TRequestToken	requestToken
	TLifeTimeInSeconds	estimatedProcessingTime,
	TRetentionPolicyInfo	retentionPolicyInfo,
	TSizeInBytes	sizeOfTotalReservedSpace, // best effort
	TSizeInBytes	sizeOfGuaranteedReservedSpace,
	TLifeTimeInSeconds	lifetimeOfReservedSpace,
	TSpaceToken,	spaceToken,
	TReturnStatus	returnStatus

Notes:

- Asynchronous space reservation may be necessary for some SRMs to serve many concurrent requests. In such case, request token must be returned, and space token must not be assigned and returned until space reservation is completed. If the space reservation can be done immediately, request token must not be returned.

- When asynchronous space reservation is necessary, the returned status code should be SRM_REQUEST_QUEUED.
- Input parameter *expectedFileSize* is a hint that SRM server can use to reserve consecutive storage sizes for the request. At the time of space reservation, if space accounting is done only at the level of the total size, this hint would not help. In such case, the expected file size at the time of PrepareToPut will describe how much consecutive storage size is needed for the file. However, some SRMs may get benefits from these hints to make a decision to allocate some blocks in some specific devices.
- Optional input parameter *storageSystemInfo* is needed in case the underlying storage system requires additional security information.
- SRM may return its default space size and lifetime if not requested by the client. SRM may return SRM_INVALID_REQUEST if SRM does not support default space sizes.
- If input parameter *sizeOfTotalSpaceDesired* is not specified, the SRM will return its default space size.
- Output parameter *estimateProcessingTime* is used to indicate the estimation time to complete the space reservation request, when known.
- Output parameter *sizeOfTotalReservedSpace* is in best effort bases. For guaranteed space size, *sizeOfGuaranteedReservedSpace* should be checked. These two numbers may match, depending on the storage systems.
- Output parameter *spaceToken* is a reference handle of the reserved space.

srmGetStatusOfReserveSpace

This function is used to check the status of the previous request to *srmReserveSpace*, when asynchronous space reservation was necessary with the SRM. Request token must have been provided in response to the *srmReserveSpace*.

In:	TUserID	authorizationID,
	TRequestToken	requestToken
Out:	TLifeTimeInSeconds	estimatedProcessingTime,
	TRetentionPolicyInfo	retentionPolicyInfo,
	TSizeInBytes	sizeOfTotalReservedSpace,
	TSizeInBytes	sizeOfGuaranteedReservedSpace,
	TLifeTimeInSeconds	lifetimeOfReservedSpace,
	TSpaceToken,	spaceToken,
	TReturnStatus	returnStatus

Notes:

- If the space reservation is not completed yet, *estimateProcessingTime* is returned when known. The returned status code in such case should be SRM_REQUEST_QUEUED.
- See notes for *srmReserveSpace* for descriptions for output parameters.

srmGetSpaceMetaData

This function is used to get information of a space. Space token has to be provided, and space tokens are returned upon a completion of a space reservation through *srmReserveSpace* or *srmGetStatusOfReserveSpace*.

```
typedef     struct { TSpaceToken            spaceToken,
                     TRetentionPolicyInfo   retentionPolicyInfo,
                     Boolean                isValid,
                     TUserID                owner,
                     TSizeInBytes           totalSize,        // best effort
                     TSizeInBytes           guaranteedSize,
                     TSizeInBytes           unusedSize,
                     TLifeTimeInSeconds     lifetimeAssigned,
                     TLifeTimeInSeconds     lifetimeLeft
                   } TMetaDataSpace
```

- *TMetaDataSpace* is used to describe properties of a space, and is used as an output parameter in *srmGetSpaceMetaData*.
- *retentonPolicyInfo* is added to indicate the information about retention policy and access latency that the space is assigned. *retentionPolicyInfo* is requested and assigned at the time of space reservation through *srmReserveSpace* and *srmGetStatusOfReserveSpace*.

Details:

```
In:   TUserID              authorizationID,
      TSpaceToken[]        arrayOfSpaceToken

Out:  TMetaDataSpace[]     arrayOfSpaceDetails
      TReturnStatus        returnStatus
```

srmChangeRetentionPolicy

This function is used to change the retention policy of files to another retention policy by specifying source and target space tokens. All files specified by SURLs that are associated with the space token will have a new space token. New space token may be acquired from *srmReserveSpace*. Asynchronous operation may be necessary for some SRMs, and in such case, request token is returned for later status inquiry. There is no default behavior when source or target space token is not provided. In such case, the request will be rejected, and the return status must be SRM_INVALID_REQUEST.

```
In:   TUserID              authorizationID
```

	TSURLInfo[]	arrayOfSURLs
	TSpaceToken	sourceSpaceToken
	TSpaceToken	targetSpaceToken
Out:	TRequestToken	requestToken
	TLifeTimeInSeconds	estimatedProcessingTime
	TReturnStatus	returnStatus

Notes:

- Asynchronous operation may be necessary for some SRMs to serve many concurrent requests. In such case, request token must be returned. If the request can be completed immediately, request token must not be returned.
- When asynchronous operation is necessary, the returned status code should be SRM_REQUEST_QUEUED.
- All files specified in *arrayOfSURLs* in the space associated with *sourceSpaceToken* will be moved to the space associated with *targetSpaceToken.*
- If any *arrayOfSURLs* are not specified, all files in the space associated with *sourceSpaceToken* may be moved to the target space associated with *targetSpaceToken*.
- If target space token is to be used, space allocation for a new space token must be done explicitly by the client before using this function.
- Space de-allocation may be necessary in some cases where source space token is associated, and it must be done by the client explicitly after this operation completes. The status can be checked by srmGetStatusOfChangeRetentionPolicy.

srmGetStatusOfChangeRetentionPolicy

This function is used to check the status of the previous request to *srmChangeRetentionPolicy,* when asynchronous operation was necessary in the SRM. Request token must have been provided in response to the *srmChangeRetentionPolicy*.

In:	TUserID	authorizationID,
	TRequestToken	requestToken
Out:	TLifeTimeInSeconds	estimatedProcessingTime,
	TReturnStatus	returnStatus

N**otes:**

- If changing retention policy is not completed, *estimateProcessingTime* is returned when known. The returned status code in such case should be SRM_REQUEST_QUEUED.

srmExtendFileLifeTimeInSpace

This function is used to extend lifetime of the files in a space.

In:	TSpaceToken	spaceToken,
	TSURL[]	SURLs,
	TUserID	authorizationID,
	TLifeTimeInSeconds	newLifeTime
Out:	TReturnStatus	returnStatus,
	TLifeTimeInSeconds	newTimeExtended

Notes:

- When *spaceToken* is provided, the lifetime of the file copy of the SURLs in the space associated with the space token will be extended.
- *newLifeTime* is relative to the calling time. Lifetime will be set from the calling time for the specified period.
- The number of lifetime extensions maybe limited by SRM according to its policies.
- If original lifetime is longer than the requested one, then the requested one will be assigned.
- If *newLifeTime* is not specified, the SRM can use its default to assign the *newLifeTime*.

2. In the Directory Functions:

Summary:

- srmRemoveFiles has an optional request token, instead of required request token.
- srmRemoveFiles has an optional space token to remove "copies" (or "states") of files in a specific space.
- TMetaDataPathDetail includes the assigned retention policy, and includes an indication of files being located online, nearline, or both.
- srmLs has TSURL and TStorageSystemInfo separately from the previously combined TSURLInfo as input parameters.

srmRm

This function will remove SURLs (the name space entries) in the storage system. Difference from *srmRemoveFiles* is that *srmRemoveFiles* removes only previously requested "copies" (or "state") of the SURL, and *srmRemoveFiles* shall not remove SURLs or the name space entries. If any files are not released yet, this function will release them before removing SURLs.

In:	TUserID	authorizationID,

	TSURLInfo[]	arrayOfSURLs
Out:	TReturnStatus	returnStatus,
	TSURLReturnStatus[]	arrayOfFileStatus

srmRemoveFiles

This function will be used to remove previously requested files (online/nearline "copies" or "states") specified by SURLs, through *srmPrepareToGet* and *srmBringOnline*. This function must not remove the SURLs, but only the "copies" or "states" of the SURLs. *srmRm* must be used to remove SURLs.

In:	TUserID	authorizationID
	TSURLInfo[]	arrayOfSURLs
	TRequestToken	requestToken
	TSpaceToken	spaceToken
Out:	TReturnStatus	returnStatus,
	TSURLReturnStatus[]	arrayOfFileStatus

Notes:

- When input parameter *requestToken* is provided, SRM will remove only the "copies" (or "state") of the *SURLs* associated with the request token.
- When input parameter *spaceToken* is provided, SRM will remove only the "copies" (or "state") of the *SURLs* associated with the space token.
- When input parameter *requestToken* and *spaceToken* are provided, SRM will verify if files associated with the request token belongs to the space associated with the space token, and SRM will remove the "copy" (or "state") of the file. *srmRemoveFiles* must not remove the SURL, the namespace entry. *srmRm* must be used for such purpose.
- It has the effect of a release on the "copy" (or "state") of the file before being removed.

srmLs

This function is used to get information of a file.

typedef	struct {TUserID	userID,
	TPermissionMode	mode
	} **TUserPermission**	

- TUserID may represent the associated client's Distinguished Name (DN) instead of unix style login name. VOMS role may be included.

```
typedef     struct {TGroupID          groupID,
                    TPermissionMode   mode
                } TGroupPermission
```

- TGroupD may represent the associated client's Distinguished Name (DN) instead of unix style login name. VOMS role may be included.

enum **TFileLocality** { ONLINE, NEARLINE, BOTH }

- Files may be located online, nearline or both. This indicates if the file is online or not, or if the file reached to nearline or not. It also indicates if there are online and nearline copies of the file.
    - The ONLINE indicates that there is a file on online cache of a storage system which is the part of the storage system, and the file may be accessed with online latencies.
    - The NEARLINE indicates that the file is located on nearline storage system, and the file may be accessed with nearline latencies.
    - The BOTH indicates that the file is located on online cache of a storage system as well as on nearline storage system.
- The type will be used to describe a file property that indicates the current location in the storage system.

```
typedef     struct {TSURL                 surl,   // both dir and file
                    TReturnStatus         status,
                    TSizeInBytes          size,    // 0 if dir
                    TOwnerPermission      ownerPermission,
                    TUserPermission[]     userPermission,
                    TGroupPermission[]    groupPermission,
                    TOtherPermission      otherPermission
                    TGMTTime              createdAtTime,
                    TGMTTime              lastModificationTime,
                    TUserID               owner,
                    TFileStorageType      fileStorageType,
                    TRetentionPolicyInfo  retentionPolicyInfo,
                    TFileLocality         fileLocality,
                    TSpaceToken[]         spaceTokens,
                    TFileType             type,    // Directory or File
                    TLifeTimeInSeconds    lifetimeAssigned,
                    TLifeTimeInSeconds    lifetimeLeft, // on the SURL
                    TCheckSumType         checkSumType,
                    TCheckSumValue        checkSumValue,
```

```
        TSURL                   originalSURL,   // if path is a file
        TMetaDataPathDetail[]   subPath         // optional recursive
    } TMetaDataPathDetail
```

- The *TMetaDataPathDetail* describes the properties of a file. It is used as an output parameter in *srmLs*.
- *retentionPolicyInfo* indicates the assigned retention policy.
- *fileLocality* indicates where the file is located currently in the system: online, nearline or both.
- *spaceTokens* as an array of *TSpaceToken* indicates where the file is currently located for the client. Only space tokens that the client has authorized to access to read the file must be returned.

Details:

In:	TUserID	authorizationID,
	TStorageSystemInfo	storageSystemInfo,
	TSURL []	surls,
	TFileStorageType	fileStorageType,
	boolean	fullDetailedList,
	boolean	allLevelRecursive,
	int	numOfLevels,
	int	offset,
	int	count
Out:	TMetaDataPathDetail[]	details,
	TReturnStatus	returnStatus

Notes:

- *Applies to both dir and file*
- *fullDetailedList=false by default.*
  - *For directories, only path is required to be returned.*
  - *For files, path and size are required to be returned.*
- *If fullDetailedList=true, the full details are returned.*
  - *For directories, path and userPermission are required to be returned.*
  - *For files, path, size, userPermission, lastModificationTime, typeOfThisFile, and lifetimeLeft are required to be returned, similar to unix command ls –l.*
- *If allLevelRecursive=true then file lists of all level below current will be provided as well.*
- *If allLevelRecursive is "true" it dominates, i.e. ignore numOfLevels. If allLevelRecursive is "false" or missing, then do numOfLevels. If numOfLevels is "0" (zero) or missing, assume a single level. If both allLevelRecursive and numOfLevels are missing, assume a single level.*
- *When listing for a particular type specified by "fileStorageType", only the files with that type will be in the output.*
- *Empty directories will be returned.*

- *We recommend width first in the listing.*
- *We recommend that list of directories come before list of files in the return array (details).*

3. In the Data Transfer Functions:

Summary:

- Client access pattern is added to indicate the possible usage pattern of the TURL.
- Client connection type is added to indicate the possible connection to the TURL.
- TTransferProtocol is added to combine the client input parameters for array of client supported transfer protocol list, client access pattern, and client connection type.
- TExtraInfo is added for additional information about the returned transfer protocol of TURL. It may indicate the properties of the transfer protocol so that the client can optimize the date transfer.

enum **TAccessPattern** { TransferMode, ProcessingMode }

- TAccessPattern will be passed as an input parameter to the srmPrepareToGet and srmBringOnline functions. It will make a hint from the client to SRM how the Transfer URL (TURL) produced by SRM is going to be used. If the parameter value is "ProcessingMode", the system may expect that client application will perform some processing of the partially read data, followed by more partial reads and a frequent use of the protocol specific "seek" operation. This will allow optimizations by allocating files on disks with small buffer sizes. If the value is "TransferMode" the file will be read at the highest speed allowed by the connection between the server and a client.

enum **TConnectionType** { WAN, LAN }

- TConnectionType indicates if the client is connected though a local or wide area network. SRM may optimize the access parameters to achieve maximum throughput for the connection type. This will be passed as an input to the srmPrepareToGet, srmPrepareToPut and srmBringOnline functions.

```
typedef     struct {TAccessPattern      accessPattern,
                    TConnectionType     connectionType,
                    string[]            arrayOfTransferProtocols
                } TTransferProtocol
```

- TTransferProtocol is used where arrayOfTransferProtocols was used previously.

- TGetFileRequest includes TAccessPattern which may conflict with the online disk type of the target space associated with target space token if provided. In this case, TAccessPattern must be ignored.

```
typedef     struct {string      key,
                    string      value,
                } TExtraInfo
```

- TExtraInfo is used where additional information is needed, such as for additional information for transfer protocols of TURLs in srmStatusOfGetRequest, srmStatusOfPutRequest, srmGetTransferProtocols, and srmPing.
- For example, when it is used for additional information for transfer protocols, the keys may specify access speed, available number of parallelism, and other transfer protocol properties.

srmPrepareToGet

This function is used to bring files online upon the client's request and assign TURL so that client can access the file. Lifetime (pinning expiration time) is assigned on the TURL. When specified target space token which must be referred to an online space, the files will be prepared using the space associated with the space token. It may be an asynchronous operation, and request token must be returned if asynchronous operation is necessary in SRM. The status may be checked through srmGetStatusOfPrepareToGet with the returned request token.

```
typedef     struct {TSURL               fromSURL,
                    TDirOption          dirOption,
                } TGetFileRequest
```

- TLifetimeInSeconds, TFileStorageType and TSpaceToken are removed from the previous TGetFileRequest. fromSURL becomes of type TSURL instead of TSURLInfo so that TStorageSystemInfo can be removed at the file level. Those removed input parameters may be provided at the request level, instead of the file level. This will make the interface and implementation simpler. It prevents different file storage types that can be requested at the same time into different target space for multiple files.

```
typedef     struct {TSURL               fromSURL,
                    TSizeInBytes        fileSize,
                    TReturnStatus       status,
                    TLifeTimeInSeconds  estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds  estimatedProcessingTime,
                    TLifeTimeInSeconds  remainingPinTime,
                    TFileStorageType    fileStorageType
```

	TSpaceToken	spaceToken,
	TTURL	transferURL
	TExtraInfo	transferProtocolInfo
} **TGetRequestFileStatus**		

- *transferProtocolInfo* of type *TExtraInfo* is added to the *TGetRequestFileStatus*. This output parameter can be used to provide more information about the transfer protocol so that client can access the TURL efficiently.
- *TSpaceToken* is added to the *TGetRequestFileStatus* to show and confirm which space is used for the file, if client provided the target space token at the time of the *srmPrepareToGet*.
- *TFileStorageType* is added to the *TGetRequestFileStatus* to show and confirm which file storage type is used for the file, if client provided the desired file storage type at the time of the *srmPrepareToGet*.

Details:

In:	TUserID	authorizationID,
	TGetFileRequest[]	arrayOfFileRequest,
	string	userRequestDescription,
	TStorageSystemInfo	storageSystemInfo,
	TLifeTimeInSeconds	totalRetryTime
	TFileStorageType	preferredFileStorageType
	TLifeTimeInSeconds	desiredLifetime,
	TSpaceToken	targetSpaceToken
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo
	TTransferProtocol	transferProtocolList
Out:	TRequestToken	requestToken,
	TReturnStatus	returnStatus,
	TGetRequestFileStatus[]	arrayOfFileStatus

- Those file level input parameters (TFileStorageType, TLifeTimeInSeconds, TSpaceToken) are now at the request level to simplify the interface and implementation.
- Array of transfer protocols is combined with access pattern and connection type in transfer protocol list as TTransferProtocol.

Notes:

- If input parameter *TSpaceToken* is provided, then the target space token must refer to online space. All requested files will be prepared into the target space.
- Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is prepared online.

- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected.
- Access latency must be ONLINE always.
- Input parameter *TAccessPattern* is provided at the request-level, and all files will have the same access pattern.
- *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferProtocol* must be ignored.
- The *userRequestDescription* is a user designated name for the request. It can be used in the *srmGetRequestID* function to get back the system assigned request tokens.
- Only pull mode is supported for file transfers that client must pull the files from the TURL within the expiration time (*remainingPinTime*).
- Input parameter *desiredLifetime* is for a client preferred lifetime (expiration time) on the prepared TURL.
- If asynchronous operation is needed, SRM assigns the *requestToken* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.
- When the file is ready and TURL is prepared, the return status code should be SRM_FILE_PINNED.
- "*retryTime*" means: if all the file transfer for this request are complete, then try previously failed transfers in this request for a total time period of "*retryTime*".
- In case that the retries fail, the return status should include an explanation of why the retries failed.
- This call may be an asynchronous (non-blocking) call. To get subsequent status and results, separate calls through *srmGetStatusOfPrepareToGet* should be made with request token.
- When the file is ready for the user, the file is implicitly pinned in the cache and lifetime will be enforced.
- The invocation of *srmReleaseFile*() is expected for finished files later on.
- The returned request token should be valid until all files in the request are released or removed.

srmPrepareToPut

This function is used to write files into the storage. Upon the client's request, SRM prepares a TURL so that client can write data into the TURL. Lifetime (pinning expiration time) is assigned on the TURL. When a specified target space token is provided, the files will be located finally in the targeted space associated with the target space token. It may be an asynchronous operation, and request token must be returned if asynchronous operation is necessary in SRM. The status may be checked through srmGetStatusOfPrepareToPut with the returned request token.

typedef struct {TSURL targetSURL , // local to SRM
TSizeInBytes expectedFileSize
} **TPutFileRequest**

- targetSURL is required.
- TLifetimeInSeconds, TFileStorageType and TSpaceToken are removed from the previous TPutFileRequest. targetSURL becomes of type TSURL instead of TSURLInfo so that TStorageSystemInfo can be removed at the file level. Those removed input parameters can be provided at the request level, instead of the file level. This will make the interface and implementation simpler. It prevents different file storage types that can be requested at the same time into different target space for multiple files.

typedef struct { TSizeInBytes fileSize,
TReturnStatus status,
TLifeTimeInSeconds estimatedWaitTimeOnQueue,
TLifeTimeInSeconds estimatedProcessingTime,
TTURL transferURL,
TSURL SURL,
TLifeTimeInSeconds remainingPinTime // on TURL
TFileStorageType fileStorageType
TSpaceToken spaceToken,
TExtraInfo transferProtocolInfo
} **TPutRequestFileStatus**

- *TSpaceToken* is added to the *TPutRequestFileStatus* to show and confirm which space is used for the file, if client provided the target space token at the time of the *srmPrepareToPut*.
- *TFileStorageType* is added to the *TPutRequestFileStatus* to show and confirm which file storage type is used for the file, if client provided the desired file storage type at the time of the *srmPrepareToPut*.
- *transferProtocolInfo* of type *TExtraInfo* is added to the *TPutRequestFileStatus* to give clients more information about the prepared transfer protocol so that client may use the information to make an efficient access to the prepared TURL through the transfer protocol.

Details:

In: TUserID authorizationID,
TPutFileRequest[] arrayOfFileRequest,
string userRequestDescription,
TOverwriteMode overwriteOption,
TStorageSystemInfo storageSystemInfo,
TLifeTimeInSeconds totalRetryTime
TLifeTimeInSeconds desiredPinLifetime, // on TURL

	TLifeTimeInSeconds	desiredFileLifetime, // on SURL
	TFileStorageType	preferredFileStorageType,
	TSpaceToken	targetSpaceToken
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo
	TTransferProtocol	transferProtocolList
Out:	TRequestToken	requestToken,
	TReturnStatus	returnStatus,
	TPutRequestFileStatus[]	arrayOfFileStatus

- Those file level input parameters (TLifeTimeInSeconds, TFileStorageType and TSpaceToken) are now at the request level to simplify the interface and implementation.
- Array of transfer protocols is combined with access pattern and connection type in transfer protocol list as TTransferProtocol.

Notes:

- *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferProtocol* must be ignored.
- Input parameter *TSpaceToken* is provided at the request-level, and all files in the request will end up in the space that is associated with the target space token.
- Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is written into the target storage system.
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected.
- Only push mode is supported for file transfers that client must "push" the file to the prepared TURL.
- Input parameter *targetSURL* in the *TPutFileRequest* has to be local to SRM. If *targetSURL* is not specified, SRM will make a reference for the file request automatically and put it in the specified user space if provided. This reference url will be returned along with the "Transfer URL".
- *srmPutDone*() is expected after each file is "put" into the prepared TURL.
- Input parameter *desiredPinLifetime* is the lifetime (expiration time) on the TURL when the Transfer URL is prepared. It does not refer to the lifetime of the SURL.
- Input parameter *desiredFileLifetime* is the lifetime of the SURL when the file is put into the storage system. It does not refer to the lifetime (expiration time) of the TURL.
- The lifetime of the SURL starts as soon as SRM get the *srmPutDone*(). If *srmPutDone*() is not provided, then the files in that space are subject to removal when the lifetime on the TURL expires or the lifetime on the space expires. The lifetime on the TURL can be found in the status of the file request as output parameter *remainingPinTime* in *TPutRequestFileStatus*.

- "*retryTime*" is meaningful only when the file destination is not a local disk, such as tape or MSS.
- In case that the retries fail, the return should include an explanation of why the retries failed.

srmCopy

This function is used to copy files from source storage sites into the target storage sites. The source storage site or the target storage site needs to be the SRM itself that the client makes the *srmCopy* request. If both source and target are local to the SRM, it performed a local copy. There are two cases for remote copies: 1. Target SRM is where client makes a srmCopy request (PULL case), 2. Source SRM is where client makes a srmCopy request (PUSH case).

1. PULL case: Upon the client's *srmCopy* request, the target SRM makes a space at the target storage, and makes a request *srmPrepareToGet* to the source SRM. When TURL is ready at the source SRM, the target SRM transfers the file from the source TURL into the prepared target storage.
2. PUSH case: Upon the client's *srmCopy* request, the source SRM prepares a file to be transferred out to the target SRM, and makes a request *srmPrepareToPut* to the target SRM. When TURL is ready at the target SRM, the source SRM transfers the file from the prepared source into the prepared target TURL. After the file transfer completes, *srmPutDone* is issued to the target SRM.

When specified target space token is provided, the files will be located finally in the targeted space associated with the space token. It may be an asynchronous operation, and request token must be returned if asynchronous operation is necessary in contacting SRM. The status may be checked through srmGetStatusOfCopy with the returned request token.

```
typedef     struct {TSURLInfo       fromSURLInfo,
                    TSURLInfo       toSURLInfo,
                    TDirOption      dirOption
                } TCopyFileRequest
```

- TLifetimeInSeconds, TFileStorageType, TSpaceToken and TOverwriteMode are removed from the previous TCopyFileRequest. Those removed input parameters can be provided at the request level, instead of the file level. This will make the interface and implementation simpler. It prevents different file storage types that can be requested at the same time into different target space for multiple files.

```
typedef     struct {TSURL           fromSURL,
                    TSURL           toSURL,
                    TSizeInBytes    fileSize,
```

	TReturnStatus	status,
	TLifeTimeInSeconds	estimatedWaitTimeOnQueue,
	TLifeTimeInSeconds	estimatedProcessingTime,
	TLifeTimeInSeconds	remainingPinTime
	TFileStorageType	targetFileStorageType
	TSpaceToken	targetSpaceToken
} **TCopyRequestFileStatus**		

- TSpaceToken is added to the TCopyRequestFileStatus to show and confirm which space is used for the file, if client provided the target space token at the time of the srmCopy.
- TFileStorageType is added to the TCopyRequestFileStatus to show and confirm which file storage type is used for the file, if client provided the target file storage type at the time of the srmCopy.

Details:

In:	TUserID	authorizationID,
	TCopyFileRequest[]	arrayOfFileRequest,
	string	userRequestDescription,
	TOverwriteMode	overwriteOption,
	Boolean	removeSourceFiles (default = false),
	TLifeTimeInSeconds	totalRetryTime
	TLifeTimeInSeconds	lifetime, // on target SURLs
	TFileStorageType	targetFileStorageType,
	TOverwriteMode	overwriteMode,
	TSpaceToken	targetSpaceToken,
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo
Out:	TRequestToken	requestToken,
	TReturnStatus	returnStatus,
	TCopyRequestFileStatus[]	arrayOfFileStatus

- Those file level input parameters (TLifeTimeInSeconds, TFileStorageType, TOverwriteMode and TSpaceToken) are now at the request level to simplify the interface and implementation.

Notes:

- Input parameter *TSpaceToken* is provided at the request-level, and all files in the request will end up in the space that is associated with the target space token.
- Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is written into the target storage system.

- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected.
- Pull mode: copy from remote location to the SRM. (e.g. from remote to MSS.)
- Push mode: copy from the SRM to remote location.
- Always release files through *srmReleaseFiles* from the source after copy is done, if source is an SRM and PULL mode was performed.
- Always issue *srmPutDone* to the target after copy is done, if target is an SRM and PUSH mode was performed.
- When *removeSourceFiles* is true, then SRM will request *srmRm* (in case the source is an SRM) or will use any other means to remove the source files on behalf of the caller after copy is done.
- Note there is no protocol negotiation with the client for this request.
- "retryTime" means: if all the file transfer for this request are complete, then try previously failed transfers in this request for a total time period of "retryTime".
- In case that the retries fail, the return status should include an explanation of why the retries failed.
- When both fromSURL and toSURL are local, perform local copy
- Empty directories are copied as well.

srmBringOnline

This function is used to bring files online upon the client's request so that client can make certain data readily available for future access. In hierarchical storage systems, it is expected to "stage" files to the top hierarchy and make sure that the files stay online for a certain period of time. When client specifies target space token which must be referred to an online space, the files will be brought online using the space associated with the space token. It may be an asynchronous operation, and request token must be returned if asynchronous operation is necessary in SRM. The status may be checked through *srmGetStatusOfBringOnline* with the returned request token.
This function is similar to srmPrepareToGet, but it does not return Transfer URL (TURL).

```
typedef     struct {TSURL                   fromSURL,
                    TSizeInBytes            fileSize,
                    TReturnStatus           status,
                    TLifeTimeInSeconds      estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds      estimatedProcessingTime,
                    TLifeTimeInSeconds      remainingPinTime,
                    TFileStorageType        fileStorageType
                    TSpaceToken             spaceToken
            } TBringOnlineRequestFileStatus
```

Details:

```
    In:     TUserID                 authorizationID,
```

	TGetFileRequest[]	arrayOfFileRequest,
	string	userRequestDescription,
	TStorageSystemInfo	storageSystemInfo,
	TLifeTimeInSeconds	totalRetryTime
	TFileStorageType	preferredFileStorageType
	TLifeTimeInSeconds	desiredLifetime, // pin time
	TSpaceToken	targetSpaceToken,
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo
	TTransferProtocol	transferProtocolList
Out:	TRequestToken	requestToken,
	TReturnStatus	returnStatus,
	string[]	supportedTransferProtocols,
	TBringOnlineRequestFileStatus[]	arrayOfFileStatus

- The difference from srmPrepareToGet is in the output parameters.
    - *supportedTransferProtocols* is an array of transfer protocols that SRM can support among the client provided transfer protocol list.
    - *TBringOnlineRequestFileStatus* resembles the *TGetFileStatus*, but does not contain TURL and TransferProtocolInfo.

Notes:

- Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is brought online.
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected.
- *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferProtocol* must be ignored.
- When *arrayOfTransferProtocols* are submitted, SRM returns those transfer protocols that SRM supports among the user-submitted transfer protocols.
- The *userRequestDescription* is a user designated name for the request. It can be used in the *srmGetRequestID* method to get back the system assigned request ID.
- Input parameter *desiredLifetime* is for a client preferred lifetime (expiration time) on the file "copies (or "states") of the SURLs that will be "brought online" into the target space that is associated with the *targetSpaceToken*.
- SRM assigns the *requestToken* at the time of asynchronous operation when necessary.
- "*retryTime*" means: if all the file transfer for this request are complete, then try previously failed transfers for this request for a total time period of "*retryTime*".
- In case that the retries fail, the return status should include an explanation of why the retries failed.
- This call may be an asynchronous (non-blocking) call. To get subsequent status and results, separate calls should be made through srmStatusOfBringOnline.

- The returned request token should be valid until all files in the request are released, removed or aborted.
- When *srmAbortRequest* is requested for *srmBringOnline* request, the request gets aborted, but those files that are brought online will remain in the space where they are brought in, and are not removed. Clients need to remove those files through *srmRemoveFiles*.

srmStatusOfBringOnlineRequest

This function is used to check the status of the previous request to *srmBringOnline,* when asynchronous operation is necessary in the SRM. Request token must have been provided in response to the *srmBringOnline*.

In:	TRequestToken	requestToken,
	TUserID	authorizationID
	TSURL[]	arrayOfFromSURLs,
Out:	TReturnStatus	returnStatus,
	TBringOnlineRequestFileStatus[]	arrayOfFileStatus

Notes:
- If *arrayOfFromSURLs* is not provided, returns status for all files in this request.

4. In the Information Discovery Functions:

Summary:
- srmGetSRMStorageInfo is added for clients to discover the system information.
- srmGetTransferProtocols is added for clients to discover the supported transfer protocols by SRM.
- srmPing is added for clients to check the status of the SRM.

srmGetSRMStorageInfo

This function is used to discover what features SRM supports and what the default values for a specific feature in SRM.

```
enum        TStorageAttributes  { SRM_STORAGE_CAPACITY,
                                  SRM_USER_STORAGE_MAX,
                                  SRM_USER_STORAGE_MIN,
```

SRM_USER_STORAGE_DEFAULT_LIFETIME,
SRM_DEFAULT_FILE_LIFETIME,
SRM_DEFAULT_FILE_STORAGE_TYPE,
SRM_DEFAULT_TURL_EXPIRATION_TIME,
SRM_ DEFAULT_ACCESS_LATENCY,
SRM_ DEFAULT_ACCESS_PATTERN,
SRM_ DEFAULT_CONNECTION_TYPE,
SRM_ DEFAULT_RETENTION_POLICY,
SRM_SUPPORTED_FILE_STORAGE_TYPE,
SRM_SUPPORTED_RETENTION_POLICY,
SRM_SUPPORTED_ACCESS_PATTERN,
SRM_SUPPORTED_CONNECTION_TYPE,
SRM_SUPPORTED_ACCESS_LATENCY,
SRM_DEFAULT_TRANSFER_PROTOCOL
}

typedef struct {
TStorageAttributes storageAttr,
string value,
string valueType,
string explanation
} **TStorageInfo**

- value - value of the *storageAttr.* When SRM supports multiple values (e.g. when SRM supports multiple retention policies), this value may contain more than one value separated by comma (,). E.g. RELICA,CUSTODIAL
- valueType - data type of the value for storageAttr in literal characters. For example, int, long, string, boolean, TRetentionPolicy, etc.
- explanation – this parameter explains what the value means to the SRM server. E.g. CUSTODIAL from TRetentionPolicy can be explained in the parameter that how the particular SRM treats this type if supported.

Details:

In:	TUserID	authorizationID,
	EnumStorageAttributes	desiredAttributes[]
Out:	TReturnStatus	returnStatus,
	TStorageInfo	storageInfo[]

Notes:

- *srmGetSRMStorageInfo* retrieves SRM storage information, such as storage capacity, client quota, default lifetime, etc.

- When output parameter, TStorageInfo is returned to the client, *storageAttr* and its *value* are required to be returned.

srmGetTransferProtocols

This function is to discover what transfer protocols are supported by the SRM.

typedef struct {
string transferProtocol,
TExtraInfo attributes
} **TSupportedTransferProtocols**

Note:
- *transferProtocol* (required): Supported transfer protocol. For example, gsiftp, http.
- *attributes*: Informational hints for the paired transfer protocol, such how many number of parallel streams can be used, desired buffer size, etc.

Details:

In:	TUserID		authorizationID,
Out:	TReturnStatus		returnStatus,
	TSupportedTransferProtocols		protocolInfo[]

Notes:
- *srmGetTransferProtocols*() returns the supported file transfer protocols in the SRM with any additional information about the transfer protocol.

srmPing

This function is used to check the state of the SRM. It works as an "are you alive" type of call.

Details:

In:	TUserID		authorizationID,
Out:	string		versionInfo
	TExtraInfo		otherInfo

Notes:

- *srmPing*() returns a string containing SRM v2.2 version number as a minimal "up and running" information. For this particular SRM v2.2 version, it must be "v2.2". Other versions may have "v1.1", "v3.0", and so on.
- Any additional information about the SRM can be provided in the output parameter *otherInfo*.